APPENDIX D

```
#
# procOrder.pl (c) 1998,1999 E-Dialog, Inc. All Rights Reserved.
#
# PURPOSE DISCUSSION
#
# program accepts a raw email stream from MS Exchange Server via export to ADO-compliant datastore
# then uses email body preprocessing and extraction rules - tailored per campaign in ./cf/ - to construct order reports.
#
#       cf files are campaign optimized:
#
#       mailingDBLayout.cf                      -> field mappings for a campaign's master mailing table
#       messageBodyDataLayout.cf                -> DSNs, field mappings, critical rule designations, order-designation bounds, synonymn table
#       messageBodyIndicateAddressChange.cf     -> rulesets for indicating ADDRESS CHANGES
#       messageBodyIndicateBuyAll.cf            -> rulesets for indicating BUY ALL state
#       messageBodyPreProcRules.cf              -> body text preprocessor configuration and rulesets
#
# program produces diagnostic and order-entry bound output files and logs outcomes of all ruleset evaluations leading to
# preparation of those output files, which include:
#
#       <$ARGV[1]>              -> summary order report file (designed for data entry, faxing, emailing to fulfillment)
#       EXC_<...>               -> (companion) summary exception report (same format as s. order report with annotations to indicate rule failures for manual review)
#       OUT_<...>               -> detailed order stream file (all fields discretely mapped and normalized for input into batch postprocessor)
#       EXC_DETAIL_<...>        -> (companion) ..exceptions..discrete fields, etc.
#       EXC_BODY_<...>          -> annotated diagnostic report containing all message bodies - sans original message - for aid in resolving exceptions manually
#
# output files are (default) tab-delimited text
#
# PROGRAM REQUIRES
#
#       Perl 5.003 interpreter later compiled for WIN32
#       MS ADO components 1.5 (2.0+ recommended)
#
# RELEASE HISTORY
#
# 981020AE: 0.5         first usage: to extract carol10 order reports
# 981204AE: 0.9         upgrade to handle carol12, multiple item selections
# 981208AE: 1.1         improve flagging
# 981211AE: 1.5         adapt to extract IDEAS@WORK add/changes
# 981222AE: 2.3.1       fix bugs in EXC_BODY reporting / enhance it
# 981222AE: 2.4         add cfPATH support
# 981223AE: 2.4.1       cfPath logging; fix synonymn (SUBJECT TOKEN) lookup <trailing \s*>
# 981223AE: 2.5         reorder reporting fields
# 981224AE: 2.5.1       reorder reporting fields
# 981225AE: 2.5.2       update SHIPPING_ADDRESS fields to reflect presence of T,B & S diagnostic fields
# 981226AE: 2.6         code review/dox; update cf to require MAILINGID
# 990106AE: 2.6.5       standardize on LOG ID for use in address change
# 990129ae: 2.7         add support for separate billing, shipping addresses
# 990201ae: 3.0         add record-recovery via ADO -> mailing database lookups
# 990202ae: 3.0.1       record-update ADO -> mailing database
# 990202ae: 3.0.2       fix leading digits bug in getDateTimeStamp ftn
# 990203ae: 3.1         fix MDB query bugs; introduce detection of SELECT && UPDATE SUCCESS; verify decision-tree based UPDATES
# 990205ae: 3.1.1       fix address3 bug; provide regression compatibility for /ADDCHANGE
# 990206ae: 3.1.2       throw .noTok as a HARD_EXC

################### INCL #####################

use Win32::OLE;

################### ENV  #####################

$GrequiredARG   = 4;
$Grevision      = '3.1.2';

if(scalar(@ARGV) < $GrequiredARG) { die "($Grevision) Usage: procOrder.pl [userName] [reportName] [cfPATH <campaignMnemonic>] [ <buyToken> | /multi | /addChange ]\n"; }

$Guser                          = $ARGV[0];
```

```
$GreportFile                    = $ARGV[1];
$GcfPath                        = $ARGV[2];
$Gsubj_buyToken                 = $ARGV[3];
$GexcReportFile                 = 'EXC_'.$GreportFile;
$GexcBODYReportFile             = 'EXC_BODY_'.$GreportFile;
$GoutFile                       = 'OUT_'.$GreportFile;
$GexceptionsFile                = 'EXC_DETAIL_'.$GreportFile;

if($ARGV[4] == 1) { $GDEBUG = 1; }      else {$GDEBUG = 0;}
if($ARGV[4] == 2) { $LGDEBUG_FTN = 1; } else {$LGDEBUG_FTN = 0; }

################### DEFN #####################

## MISC GLOBALS ##

$Gplatform                      = 'WIN';
$GeventType                     = 'INFO';
$basePath                       = '.';
$GfieldDelimiter                = "\t";
$Gspace                         = ' ';
$Gnewline                       = "\n";
$Gcomma                         = ',';
$Gpipe                          = '__';
$eventRequiresFUP               = '';
$GmultiItemOrderSwitch          = '/MULTI';
$GaddChangeSwitch               = '/ADDCHANGE';

############# OUTPUT FILE LAYOUTS ###############

## DO NOT change alpha ordering of lowercase field names (e_ .. t_)
## This is vital in the BODY field layout defn file since the rules reference the letter prefixes

## reportFile (note: EXC_reportFileName also written from this template)

        %GreportFileRecord = (

                a_LID                     => '_e_$thisLogKey',
                ab_LINE                   => '<fill-in>',
                b_RID                     => '_e_$thisRecordID',
                c_THIS_SUBJECT            => '_e_$thisSubject',
                d_THIS_BODY_BOTTOM        => '_e_$thisBodyBottom',
                e_THIS_BODY_TOP           => '_e_$thisBodyTop',
                ia_FUP_REQUIRED           => '_e_$eventRequiresFUP',
                ib_LOCATION_BUY_TOKEN     => '_e_$thisBuyStatus',
                j_REVIEW_ACTION_TAKEN     => '<review>',
                za_SELECTED_ITEMS         => '_e_$thisItemsSelectedStack',
                zd_PRIORITY               => '_e_$thisMailingID',
                zf_CUSTNO                 => '_e_($thisCustNo==-1 || $thisCustNo == 0)?"":$thisCustNo',
                zi_LISTID                 => '_e_($thisListID==-1)?"":$thisListID',
                zm_FROM_ADDRESS           => '_e_$thisFromAddress',
                zp_SPECIAL_INSTRUCTIONS => '<none>',
                zr_PHONE                  => '_e_$results{\'r_bphone\'}',
                zx_BILLING_ADDRESS        => '_e_$thisBillingAddressBlock',
                zy_SHIPPING_ADDRESS       => '_e_$thisShippingAddressBlock',
                zz_MDB_BILLING_ADDRESS    => '_e_$thisMDBbillingAddressBlock',
        );


## outFile

        %GoutFileRecord = (

                a_LID                     => '_e_$thisLogKey',
                aa_RID                    => '_e_$thisRecordID',
                ab_THIS_SUBJECT           => '_e_$thisSubject',
                ac_THIS_BODY_BOTTOM       => '_e_$thisBodyBottom',
                ad_THIS_BODY_TOP          => '_e_$thisBodyTop',
                b_LISTID                  => '_e_$thisListID',
                bb_CUSTNO                 => '_e_$thisCustNo',
                c_PRIORITY              => '_e_$thisMailingID',

                e_bfirst                => '_e_$results{\'e_bfirst\'}',
                f_bmiddle               => '_e_$results{\'f_bmiddle\'}',
                g_blast                 => '_e_$results{\'g_blast\'}',
                h_btitle                => '_e_$results{\'h_btitle\'}',
                i_bcompany              => '_e_$results{\'i_bcompany\'}',
                j_bdepartment           => '_e_$results{\'j_bdepartment\'}',
                k_baddress1             => '_e_$results{\'k_baddress1\'}',
                l_baddress2             => '_e_$results{\'l_baddress2\'}',
                m_baddress3             => '_e_$results{\'m_baddress3\'}',
                n_bcity                 => '_e_$results{\'n_bcity\'}',
                o_bstate                => '_e_$results{\'o_bstate\'}',
                p_bpostal               => '_e_$results{\'p_bpostal\'}',
                q_bcountry              => '_e_$results{\'q_bcountry\'}',
                r_bphone                => '_e_$results{\'r_bphone\'}',
                s_bfax                  => '_e_$results{\'s_bfax\'}',
                t_bemail                => '_e_$results{\'t_bemail\'}',

                ue_sfirst               => '_e_$results{\'ue_sfirst\'}',
                uf_smiddle              => '_e_$results{\'uf_smiddle\'}',
                ug_slast                => '_e_$results{\'ug_slast\'}',
                uh_stitle               => '_e_$results{\'uh_stitle\'}',
                ui_scompany             => '_e_$results{\'ui_scompany\'}',
                uj_sdepartment          => '_e_$results{\'uj_sdepartment\'}',
                uk_saddress1            => '_e_$results{\'uk_saddress1\'}',
                ul_saddress2            => '_e_$results{\'ul_saddress2\'}',
                um_saddress3            => '_e_$results{\'um_saddress3\'}',
                un_scity                => '_e_$results{\'un_scity\'}',
                uo_sstate               => '_e_$results{\'uo_sstate\'}',
                up_spostal              => '_e_$results{\'up_spostal\'}',
                uq_scountry             => '_e_$results{\'uq_scountry\'}',
                ur_sphone               => '_e_$results{\'ur_sphone\'}',
                us_sfax                 => '_e_$results{\'us_sfax\'}',
                ut_semail               => '_e_$results{\'ut_semail\'}',

                ux_FROM_ADDRESS         => '_e_$thisFromAddress',
                uy_LOCATION_BUY_TOKEN   => '_e_$thisBuyStatus',
                v_SUBJECT               => '_e_$thisSubject',
                z_FUP_REQUIRED          => '_e_$eventRequiresFUP',
                zz_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
        );

## exceptionsFile - BODY only

        %GexceptionsBODYFileRecord = (

                a_LID                   => '_e_("====================================================================================================================\n>>".$thisLogKey."<<")',
                aa_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
                ab_EXCEPT_FLAGS         => '_e_$eventRequiresFUP',
                b_RID                   => '_e_("<<".$thisRecordID.">>")',
                u_FROM_ADDRESS          => '_e_("||".$thisFromAddress."||")',
                v_SUBJECT               => '_e_("||".$thisSubject."||\n====================================================================================================================")',
                w_BODY_BELOW            => '_e_$thisFullNumberedBody',                  # full body

        );

## activityLogFile

        %GactivityLogRecord = (

                a_LID                   => '_e_$thisLogKey',
                b_user                  => '_e_$Guser."/".$GcfPath',
                c_dateStamp             => '_e_&getDateStamp(\'WIN\',$LGDEBUG_FTN)',
                e_RID                   => '_e_$thisRecordID',
                f_clientName            => '_e_$GclientName',
                g_eventType             => '_e_$eventType',
                h_eventDesc             => '_e_$eventDesc',
                z_FUP_REQUIRED          => '_e_$eventRequiresFUP',
        );


################### DATA #####################

&getKeyPos_Outfile(\%GoutFileRecord);           # identify the fieldwise position of fields in the outfile for canonicalization (use EXC file as model)

##print "--- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";

$GDSN                           = 'dsnOrderProcessing';

if ($Gplatform ne 'WIN') {$relativePath = '/logs/'} else { $relativePath = '\\logs\\';}

$GactivityLogFileName           = $basePath.$relativePath.'activityLog.tab';
$GactivityLogKeysFileName       = $basePath.$relativePath.'activityLogKeys.tab';


if ($Gplatform ne 'WIN') {$relativePath = '/db/'} else { $relativePath = '\\db\\';}

$GexcReportDBkeysFileName       = $basePath.$relativePath.'excReportDBkeys.tab';
$GexceptionsDBkeysFileName      = $basePath.$relativePath.'excDetailDBkeys.tab';
$GexceptionsBODYDBkeysFileName  = $basePath.$relativePath.'excBodyDBkeys.tab';
$GoutFileDBkeysFileName         = $basePath.$relativePath.'outFileDBkeys.tab';
$GreportFileDBkeysFileName      = $basePath.$relativePath.'reportFileDBkeys.tab';

if ($Gplatform ne 'WIN') {$relativePath = 'cf'.'/'.$GcfPath} else { $relativePath = '\\cf\\'."$GcfPath\\";}

$GbodyDataLayoutFileName        = $basePath.$relativePath.'messageBodyDataLayout.cf';
$GbodyPreProcRulesFileName      = $basePath.$relativePath.'messageBodyPreProcRules.cf';
$GbodyBuyAllFileName            = $basePath.$relativePath.'messageBodyIndicateBuyAll.cf';
$GbodyAddressChangeFileName     = $basePath.$relativePath.'messageBodyIndicateAddressChange.cf';
$GmailingDBLayoutFileName       = $basePath.$relativePath.'mailingDBLayout.cf';

################################################
#################### MAIN #####################
################################################

$MODE                   = '';
$MULTI_BUY_ON           = 0;
$ADD_CHANGE_ON          = 0;
$Gsubj_buyToken         = uc($Gsubj_buyToken);

$MODE = "<TOKEN = |$Gsubj_buyToken|>";
if($Gsubj_buyToken =~ /\/MULTI/)        { $MULTI_BUY_ON = 1; $MODE .= '<MULTI BUY ON>';}
if($Gsubj_buyToken =~ /\/ADDCHANGE/)    { $ADD_CHANGE_ON = 1; $MODE .= '<ADDCHANGE ON>'; }

print "[procOrder R.$Grevision] $MODE\n".($GDEBUG?"\n<DEBUG: $GDEBUG:$LGDEBUG_FTN>\n\n":"\n");

## open CF files

print "** Opening: $GbodyPreProcRulesFileName...\n";
        @GPreProcRules                          = &loadFile_SCALAR($GbodyPreProcRulesFileName);

print "** Opening: $GbodyBuyAllFileName...\n";
        @GBuyAllIndicators                      = &loadFile_SCALAR($GbodyBuyAllFileName);

print "** Opening: $GbodyAddressChangeFileName...\n";
        @GAddressChangeIndicators               = &loadFile_SCALAR($GbodyAddressChangeFileName);

print "** Opening: $GbodyDataLayoutFileName...\n";
        %LAYOUT                                 = &HASHloadFieldMap($GbodyDataLayoutFileName,$LGDEBUG_FTN);

unless($ADD_CHANGE_ON) {

        print "** Opening: $GmailingDBLayoutFileName...\n";
        %MDBLAYOUT = &HASHloadFieldMap($GmailingDBLayoutFileName,$LGDEBUG_FTN);
```

```
}

## open I/O

if (!open(FH_log, ">>$GactivityLogFileName"))           {$eventType = 'ABORT'; $eventDesc = "MAIN::Cannot open GactivityLogFileName: \[$fileName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
if (!open(FH_out, ">$GoutFile"))                        {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GoutFile: \[$GoutFile\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
if (!open(FH_report, ">$GreportFile"))                  {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GreportFile: \[$GreportFile\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
if (!open(FH_excReport, ">$GexcReportFile"))            {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excReportFile: \[$GexcReportFile\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
if (!open(FH_excBODYReport, ">$GexcBODYReportFile"))    {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excBODYReportFile: \[$GexcBODYReportFile\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
if (!open(FH_exc, ">$GexceptionsFile"))                 {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GexceptionsFile: \[$GexceptionsFile\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}

## load %results fieldName keys from layout cf file (now in %LAYOUT)

@fieldKeys              = undef;
@fieldKeys              = &getFieldKeys(\%LAYOUT);

        ##print "--debug--\n"; foreach(@fieldKeys) { print "**".$_."| \n"; }

$numberRequiredFields   = scalar(@fieldKeys);

if(!defined($LAYOUT{'orderProcessingDSN'})) { $LAYOUT{'orderProcessingDSN'} = $GDSN; }

print "** Loading daily order data source: $LAYOUT{'orderProcessingDSN'}...\n";
@DB                     = &read_DSNtoAOH($LAYOUT{'orderProcessingDSN'},$LGDEBUG_FTN);

$totalRecords           = $#DB +1;
$GclientName            = $LAYOUT{'thisClientName'};

## startup OK event

$eventType = '_x_INFO'; $eventDesc = "procOrder R.$Grevision START OK - USER: $Guser/$GcfPath CLIENT: $GclientName  opDSN ($totalRecords): $LAYOUT{'orderProcessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thisMailingTable'}  ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

print "\n\n** Status...\n\n";
print "USER:\t\t$Guser \n";
print "CLIENT:\t\t$GclientName\n";
print "cfPATH:\t\t$GcfPath\n\n";
print "TOTAL RECORDS \t\t -> \"$LAYOUT{'orderProcessingDSN'}\":  $totalRecords\n";
unless($ADD_CHANGE_ON) { print "MAILING TABLE SOURCE \t -> \"$LAYOUT{'thisMailingDSN'} -> $LAYOUT{'thisMailingTable'}\"\n\n"; }
print "\n(Note: if this information is not correct, hit <CNTRL> - C to abort and correct <you have 5 seconds>)\n\n"; sleep 5;

print "** Begin processing...\n\n";

## flag duplicate e-mail addresses in order stream

%dupsMap = &checkDupsReturnMap(\@DB,$GDEBUG);

###################
#### main loop ####
###################

$excpCount              = 0;
$warnCount              = 0;
$outFileCount           = 0;
$exceptionsFileCount    = 0;
$excReportFileCount     = 0;
$reportFileCount        = 0;
$FUPcount               = 0;
$reportLineCount        = 0;

for $mainCount (0 .. $#DB) {

```

```perl
$thisRecordID                   = $mainCount+1;
$eventRequiresFUP               = '';
$RECORD_VALID                   = 1;
$thisBuyStatus                  = '';
$thisShippingAddressBlock       = undef;
$thisBillingAddressBlock        = undef;
$thisMDBbillingAddressBlock     = undef;
$thisFullNumberedBody           = "\n";

## get relevant fields, normalized, this record

$thisFromAddress        = &normalizeSpacing(lc($DB[$mainCount]{'FromAddress'}));
$thisFromAddress        =~ s/[\t\r]/$Gspace/g;

$thisFromName           = &normalizeSpacing($DB[$mainCount]{'FromName'});
$thisFromName           =~ s/[\t\r]/$Gspace/g;

$thisSubject            = &normalizeSpacing(uc($DB[$mainCount]{'Subject'}));
$thisSubject            =~ s/[\t\r]/$Gspace/g;

$thisBody               = uc($DB[$mainCount]{'Body'});
$thisBody               =~ s/[\t\r]/$Gspace/g;

## split BODY into discrete lines

@BODYlines              = undef;
@topLines               = undef;
@bottomLines            = undef;

push(@BODYlines, $1) while ($thisBody =~ s/^([^\012\015]*)(\012\015?|\015\012?)//);

## skip lines that are the quoted BODY unless they're the order form -- example delimiters:
## $BODYstartToken = 'HARVARD BUSINESS SCHOOL PUBLISHING';
## $BODYendToken = 'FIRST NAME\:';

$BODYstartToken         = $LAYOUT{'startDelimiter'};
$BODYendToken           = $LAYOUT{'endDelimiter'};
$BODYstartToken         =~ s/[\t\n\r]//g;
$BODYendToken           =~ s/[\t\n\r]//g;
$startToken_ON          = 0;
$endToken_ON            = 0;
$thisBody               = undef;
$thisBodyTop            = undef;
$thisBodyBottom         = undef;
$lastLineExcluded       = undef;

for($count = 0; $count <= $#BODYlines; $count++) {

        $line                   = $BODYlines[$count];
        $thisFullNumberedBody   .= ('['.$count."] \t".$line."\n");

        unless($startToken_ON || $line =~ /^\>+/) {unless ($line !~ /\S+?/) {$thisBodyTop .= ($line.$Gpipe)}; push(@topLines,('['.$count.']  |'.$line.'|')
); }

                if(($line =~ /$BODYstartToken/) .. ($line =~ /$BODYendToken/)) {

                        if(!$startToken_ON && ($line =~ /$BODYstartToken/))     {$startToken_ON = 1;}
                        if(!$endToken_ON && ($line =~ /$BODYendToken/))         {$endToken_ON = 1; $thisBody .= ($line.$Gnewline); unless($line =~ /^\>+/
|| $line !~ /\S+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe)} }

                next;
                }

        ## ignore bottom lines if part of the original -> indicators:  ">" || "[" || "]"

        if($startToken_ON && $endToken_ON) { unless($line =~ /^\>+/ || $line !~ /\S+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe); push(@bo
ttomLines, ('['.$count.']  |'.$line.'|'));} }
        $thisBody .= ($line.$Gnewline);
```

```

$thisBodyTop    =~ s/[\t\r]/__/g;
$thisBodyBottom =~ s/[\t\r]/__/g;

                if($GDEBUG) { print "\n\n\n** DEBUG__THISBODY=".$thisBody."\n\n\n";}

## log/print warning if cannot find FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID || if bad match yields the same for CUSTNO|LISTID

        $thisCustNo             =  -1;
        $thisListID             =  -1;
        $thisMailingID          =  -1;
        $thisDialogID           =  -1;
        $LISTCID_EXCEPT_ON      =   0;
        $MAILINGID_EXCEPT_ON    =   0;

        if($thisBody !~ /\[\[([^|]*?)\|([^|]*?)\|([^|]*?)\|?([^|]*?)\]\]/) {

                $LISTCID_EXCEPT_ON = 1;

                $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   UNMATCHED FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID";
&logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
                $eventType = 'INFO'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   >>> ATTEMPTING SUBJECT LINE MAILING ID RECOVERY >>>"; &logEvent(
\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

## test to see if there is a cf file synonymn (**,++, etc) for the MAILING ID which can be recovered in the absence of the footer token

                if($thisMailingID == -1) { $thisMailingID = &subjectSynonymnLookup($thisSubject,\%LAYOUT); }
                if($thisMailingID == -1) {

                        $RECORD_VALID=0;
                        $MAILINGID_EXCEPT_ON = 1;
                        $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   MAILING ID RECOVERY WAS NOT SUCCESSFUL"; &logEvent(\*F
H_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

                if($thisMailingID != -1) {

                        $eventType = 'INFO'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   <<< MAILING ID RECOVERY SUCCESSFUL USING: $thisSubjectMa
ilingIDToken -> $LAYOUT{$thisSubjectMailingIDToken}"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }
        }

        else {

                if($GDEBUG) { print "***DEBUG_  element1= $1 __ element2= $2 ___ element3= $3 ___ element4= $4\n"; sleep 1; }

                $element1 = $1; $element2 = $2; $element3 = $3; $element4 = $4;

                $thisCustNo             = ($element1 =~ /\S+?/) ? $element1 : -1;
                $thisListID             = ($element2 =~ /\S+?/) ? $element2 : -1;
                $thisMailingID          = ($element3 =~ /\S+?/) ? $element3 : -1;
                $thisDialogID           = ($element4 =~ /\S+?/) ? $element4 : -1;

                if ($thisCustNo == -1 && $thisListID == -1) {

                        $LISTCID_EXCEPT_ON = 1;
                        $MAILINGID_EXCEPT_ON = 1;
                        $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   BAD MATCH:  CUSTNO, LISTID"; &logEvent(\*FH_log, $even
tType, $eventDesc, $Guser, $LGDEBUG_FTN);
                }

                ## log/print INFO if cannot find mailingID

                if($thisMailingID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   NO MAILING ID MATCH"; &logEvent(\*FH_log, $eventTyp
e, $eventDesc, $Guser, $LGDEBUG_FTN);}

                ## log/print INFO if cannot find dialogID

                if($thisDialogID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   NO DIALOG ID MATCH"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
        }

## check for /multi || quoted BUY TOKEN (eg: "YES")

        $buyTokenPosition       = 'none';
        $thisItemsSelectedStack = undef;
        @thisItemsSelected      = undef;

        unless($ADD_CHANGE_ON) {

                ## check for buy tokens, eg: YES in SUBJECT, TOP, BOTTOM || letter items: ABC || A,B,C || A-C

                        $buyTokenPosition = &detectBuyTokens($MULTI_BUY_ON,$Gsubj_buyToken,$thisSubject,$thisMailingID,$thisBodyTop,$thisBodyBottom,\%LAYOUT,\@topLines,\@bottomLines,$GDEBUG);

                ## check for no-match case and except

                        if($buyTokenPosition eq 'none') { $eventRequiresFUP .= '.noTok' ; $FUPcount++; $eventType = 'ERR_FUP'; $warnCount++; $eventDesc = " \<".$thisRecordID."\>   SUBJECT, TOP, BOTTOM -> NO BUY TOKEN MATCH ($Gsubj_buyToken)"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN); }

                ## if match and MULTI_BUY_ON, prepare order items listing

                        if($buyTokenPosition ne 'none' && $MULTI_BUY_ON) {

                                $thisItemsSelectedStack = &extractDedupItemsSelectedStack(@thisItemsSelected);

                        }
        }

        ################
        ## INNER LOOP ##
        ################

        %results                = {};
        $countFields            = 0;
        $nullCriticalFields     = 0;
        $ADDRESS_ADEQUATE       = 1;

## process body fields by key; keys == standard field names; values == this BODY's field names mapped thereon

        foreach (@fieldKeys) {

        $key = $_;

        ## print "--DEBUG-- key = |$key| \n";

                $thisLayoutFieldName = uc($LAYOUT{$key});
                $thisLayoutFieldName =~ s/\s+$//;

                ##################################
                ###### MAIN MATCHING BLOCK ########
                ##################################

                $MATCH_DATA_ON = 0;
                $RULE = -1;

                $expectedMatch          = $thisLayoutFieldName.'\:.*?\[([^\[]+?)\]';
                $noBracketsMatch        = $thisLayoutFieldName.'\:.*?([^\012\015]+?)';

                CASE:  {

                        TRY_EXPECTED_MATCH:     if($thisBody =~ /$expectedMatch/)               { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 1; last CASE; }
                        TRY_NO_BRAKETS_MATCH:   if($thisBody =~ /$noBracketsMatch/)             { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 2; last CASE; }

                }

## A MATCH WAS FOUND, THIS KEY ... push into results set

                if($MATCH_DATA_ON) {

                        $countFields++;
                        $matchElement  =~ s/[^\w\s\-\.\/\#\!\@]/$Gspace/g;
                        $results{$key} =  &normalizeSpacing($matchElement);

## print "--DEBUG-- result = |$results{$key}| \n";

## build shipping and billing address blocks

                        CASE: {

                                if($key =~ /u[a-z]_/) {

                                        ## construct SHIPPING ADDRESSBLOCK for report

                                        if ($key =~ /[hijklmq]\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
 }
                                        if ($key =~ /[efgnop]\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
 }

                                last CASE;
                                }

                                if($key =~ /[^u]_/) {

                                        ## construct BILLING ADDRESSBLOCK for report

                                        if ($key =~ /[hijklmq]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
}
                                        if ($key =~ /[efgnop]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
}

                                last CASE;
                                }

                        }

                        ## check for null critical fields: any absence here will throw exception, unless CUSTNO

## print "--debug-- |$LAYOUT{'criticalFields'}| \n";sleep 1;

                        if ($key =~ /[$LAYOUT{'criticalFields'}]\_/ && $results{$key} !~ /\S+?/ && $thisCustNo < 1) {

                                $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
                                $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'.badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
tDesc = " \<".$thisRecordID."\>   CRITICAL STANDARD FIELD_VALUE IS NULL: \[$thisLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $L
GDEBUG_FTN);

                        }

                }

## ALL MATCHES FAILED, THIS KEY

                if(!$MATCH_DATA_ON) {

                        $results{$key} = undef;
                        $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   STANDARD FIELD_NAME=FIELD_VALUE PAIR UNPARSABLE: \[$th
isLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                # check to see is unparsable field is a critical field
                # critical fields -> from CF file: any absence here will throw exception

```

```perl
                    if ($key =~ /[$LAYOUT{'criticalFields'}]\_/) {

                            $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
                            $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'.badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
tDesc = " \<".$thisRecordID."\>   UNPARSABLE STANDARD FIELD_NAME=FIELD_VALUE PAIR IS CRITICAL: \[$thisLayoutFieldName\]"; &logEvent(\*FH_log, $eventType,
$eventDesc, $Guser, $LGDEBUG_FTN);
                    }
            }

        }

        #######################
        #### END INNER LOOP ###
        #######################

######################
### SUMMARY RULES  ###
######################

## throw EX_ LISTNO/CUSTNO error exception if address is not OK && !phone, depending on CID_LID_REQUIRED from cf file

        if($LAYOUT{'CID_LID_REQUIRED'}) {

                if($LISTCID_EXCEPT_ON) {

                        $RECORD_VALID = 0;
                        $eventRequiresFUP .= (($eventRequiresFUP =~ /chkCLID/)?'':'.chkCLID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDes
c = " \<".$thisRecordID."\>   EITHER CID || LID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

        }

        if(!$LAYOUT{'CID_LID_REQUIRED'}) {

                if($LISTCID_EXCEPT_ON && !($ADDRESS_ADEQUATE && $results{'r\_phone'} !~ /\S+?/)) { $RECORD_VALID = 0; }

        }

## throw EX_ MAILINGID exception if cf file says it's required

        if($LAYOUT{'MAILINGID_REQUIRED'}) {

                if($MAILINGID_EXCEPT_ON) {

                        $RECORD_VALID = 0;
                        $eventRequiresFUP .= (($eventRequiresFUP =~ /chkMID/)?'':'.chkMID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc
= " \<".$thisRecordID."\>   MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

        }

## throw INFO if less than number required

        if(($countFields < $numberRequiredFields)  && $thisCustNo < 1) {

                $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   (OVERALL) NOT ALL STANDARD FIELDS FOUND: $countFields ($numberRequiredFields)"
; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
        }

## tag duplicates for FUP

if($dupsMap{$mainCount} =~ /\#\#\#.+\#\#\#/) { $eventRequiresFUP .= (($eventRequiresFUP =~ /dupAd/)?'':'.dupAd'); $FUPcount++; }

## catch thisBodyTop and thisBodyBottom blank yet address still extracted adequately (eg happens when cut & paste responses between the body delimiters)

if($RECORD_VALID && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/) {

        $RECORD_VALID = 0;
        $eventType = 'EX_ERR'; $eventRequiresFUP .= '.chkBody' ; $FUPcount++; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   CUSTOMER RESPONSE (Top,
 Bottom) NOT FOUND WHEN EXPECTED"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

}

################################################################
#### RECORD INVALID: Attempt recovery from mailing database ####
################################################################

$RECORD_RECOVERED = 0;

if(!$RECORD_VALID && !$ADD_CHANGE_ON) {

        $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   >>> BAD RECORD: ATTEMPTING RECOVERY-LOOKUP USING \"$thisFromAddress\", $LAYOUT{'thisMa
ilingDSN'} -> $LAYOUT{'thisMailingTable'}"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

        %recoveredResults = undef;
        %recoveredResults = &querySELECT_DB(\%LAYOUT,'EMAIL',$thisFromAddress,$LGDEBUG_FTN);

## foreach(keys(%recoveredResults)) { print "=debug== key = $_ ____ value = $recoveredResults{$_} \n"; }

        CASE: {

                if($recoveredResults{'_exit_'} < 1) {

                        $RECORD_VALID = 0;
                        $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB') ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   MAILING TABLE RECOVERY-LOOKUP FAILED ON \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_F
TN);

                last CASE;
                }

                if($recoveredResults{'_exit_'} == 1) {

                        $RECORD_RECOVERED = 1;
                        $RECORD_VALID = 1;
                        $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /goodMDB/)?'':'.goodMDB'); $FUPcount++; $excpCount++; $eventDe
sc = " \<".$thisRecordID."\>   <<< MAILING TABLE RECOVERY-LOOKUP SUCCESSFUL USING \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guse
r, $LGDEBUG_FTN);

                        ## construct $thisMDBbillingAddress

                        foreach $key(sort(keys(%MDBLAYOUT))) {

                                $value = $MDBLAYOUT{$key};

                        ## print "**DEBUG key = $key ___ value = $value \n";

                                if ($key =~ /[hijklmq]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$va
lue}.$Gpipe):''; }
                                if ($key =~ /[efgnop]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$val
ue}.$Gspace):''; }

                        }

                        ## restore mailingID and CID, if blank

## print "**DEBUG mailingID = $recoveredResults{'ED_MAILINGID'} __ custNo = $recoveredResults{'CID'} __ current custno = |$thisCustNo|\n";

                        if($thisMailingID !~ /\S+?/ || $thisMailingID == -1)    { $thisMailingID        = $recoveredResults{'ED_MAILINGID'}; }
                        if($thisCustNo !~ /\S+?/ || $thisCustNo == -1)          { $thisCustNo           = $recoveredResults{'CID'}; }

                last CASE;
                }

        }

```

```
}

#### add an empty flag where SHIPPING ADDRESS is returned blank

if($thisShippingAddressBlock !~ /\S+?/) { $thisShippingAddressBlock = '{ SAME }'; }
if($thisMDBbillingAddressBlock !~ /\S+?/) { $thisMDBbillingAddressBlock = '{ N/A }'; }

#### tag address block exceptions with diagnostic ques

if(!$RECORD_VALID) {

        CASE: {

                if($thisBillingAddressBlock !~ /\S+?/ && ($thisBodyTop =~ /\S+?/ || $thisBodyBottom =~ /\S+?/))         { $thisBillingAddressBlock = '=NO_
VALID_ADD/INSPECT='; last CASE;}
                if($thisBillingAddressBlock !~ /\S+?/ && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/)          { $thisBillingAddressBlock = '{ BO
DY_EMPTY/INSPECT }'; last CASE; }
                if($thisBillingAddressBlock =~ /\S+?/ && !$ADDRESS_ADEQUATE)                                           { $thisBillingAddressBlock = 'CRIT
_FLD_NUL/INSPECT ->|'.$thisBillingAddressBlock.'|'; last CASE; }

                $thisBillingAddressBlock = '=UNKNOWN_ERR/INSPECT=|'.$thisBillingAddressBlock; last CASE;

        }

$eventRequiresFUP = ($eventRequiresFUP !~ /\S+?/) ? '.inspect' : $eventRequiresFUP;
}

##########################################
## write valid record data to outfiles ##
##########################################

$ED_ORDER_VALUE = 1;
$EXCEPTION_TYPE = 'OK';
$UPDATE_SUCCESSFUL = 0;
$THIS_KEY_FIELD = undef;
$THIS_KEY_FIELD_VALUE = undef;

        if($RECORD_VALID) {

                unless($ADD_CHANGE_ON) {

                        CASE: {

                                if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisFromAddress,$THIS_KEY_FIELD = "EMAIL","ED_ORDER",$ED_ORDER_VALUE,
"ED_MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; };
                                if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisCustNo,$THIS_KEY_FIELD = "CID","ED_ORDER",$ED_ORDER_VALUE,"ED_MAI
LACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN))        { $UPDATE_SUCCESSFUL=1; last CASE; };
                                if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisListID,$THIS_KEY_FIELD = "ED_LID","ED_ORDER",$ED_ORDER_VALUE,"ED_
MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN))     { $UPDATE_SUCCESSFUL=1; last CASE; };

                        }

                }

                if(!$UPDATE_SUCCESSFUL) {

                        $RECORD_VALID = 0;
                        $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badUpMDB/)?'':'.badUpMDB') ; $FUPcount++; $excpCount++; $event
Desc = " \<".$thisRecordID."\>   (MANUAL UPDATE REQUIRED) MAILING TABLE UPDATE FAILED ON ALL KEYS - LAST TRY: \"$THIS_KEY_FIELD -> $THIS_KEY_FIELD_VALUE\"
"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

        }

        if(!$RECORD_VALUE && !$UPDATE_SUCCESSFUL) {

                $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /noUpMDB/)?'':'.noUpMDB') ; $FUPcount++; $excpCount++; $eventDesc = " \
<".$thisRecordID."\>   INVALID RECORD: MAILING TABLE NOT UPDATED"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
```

```

        }

#### Buy tokens not found and not ADDRESS_CHANGE; move to EXC report

        unless($ADDRESS_CHANGE) { if($buyTokenPosition eq 'none') { $RECORD_VALID = 0; $ED_ORDER_VALUE = 2; } }

        if($RECORD_VALID) {

                $reportLineCount++;

                ## update MAILING TABLE


                ## write out files

                &writeRecord(\*FH_out,$GfieldDelimiter,($thisKey=&getNextDBkey_return($GoutFileDBkeysFileName,$LGDEBUG_FTN)),\%GoutFileRecord,'EXTENDED',$LGDEBUG_FTN);
                &writeRecord(\*FH_report,$GfieldDelimiter,($thisKey=&getNextDBkey_return($GreportFileDBkeysFileName,$LGDEBUG_FTN)),\%GreportFileRecord,'NORMAL', $LGDEBUG_FTN);

                ## include ALL BODYs in the EXC_BODY file for potential review

                if($eventRequiresFUP =~ /\S+?/) { $EXCEPTION_TYPE = 'SOFT-EXC'; }

                        &writeRecord(\*FH_excBODYReport,$GfieldDelimiter,$thisBODYlogKey=&getNextDBkey_return($GexceptionsBODYDBkeysFileName,$LGDEBUG_FTN),\%GexceptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);

        $outFileCount++;$reportFileCount++;
        }


############################################
## write exception record data to outfiles ##
############################################

        if(!$RECORD_VALID) {

                $EXCEPTION_TYPE = 'HARD-EXC';

                &writeRecord(\*FH_exc,$GfieldDelimiter,$thisKey=&getNextDBkey_return($GexceptionsDBkeysFileName,$LGDEBUG_FTN),\%GoutFileRecord,'EXTENDED',$LGDEBUG_FTN);
                &writeRecord(\*FH_excReport,$GfieldDelimiter,$thisKey=&getNextDBkey_return($GexcReportDBkeysFileName,$LGDEBUG_FTN),\%GreportFileRecord,'NORMAL',$LGDEBUG_FTN);
                &writeRecord(\*FH_excBODYReport,$GfieldDelimiter,$thisBODYlogKey=&getNextDBkey_return($GexceptionsBODYDBkeysFileName,$LGDEBUG_FTN),\%GexceptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);

        $exceptionsFileCount++; $excReportFileCount++;
        }

print "\n";
}

######################
#### end main loop ####
######################

## print STDOUT exit state

$dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);

print "\n** [procOrder version $Grevision] Done!  $totalRecords records processed by $Guser/$GcfPath for client: $GclientName **\n\n";
print "SUMMARY REPORT for: ".$dateNow."\n\n";
print "FILES WRITTEN \n";
printf  "\n%10d",$reportFileCount;
        print  " records written to REPORT FILE: $GreportFile";
printf  "\n%10d",$excReportFileCount;
        print  " records written to EXC_REPORT FILE: $GexcReportFile";
printf  "\n%10d",$outFileCount;
```

```
        print  " records written to OUTPUT FILE (all fields): $GoutFile";
printf  "\n%10d",$exceptionsFileCount;
        print  " records written to EXCEPTIONS FILE (all fields): $GexceptionsFile";
print "\n\nLOGGING\n\n";
print "All rule failure events for all time logged in detail to: $GactivityLogFileName\n";
printf  "\n%10d",$totalRecords;
print " BODYs for this session are keyed, categorized in: $GexcBODYReportFile, for review in an editor\n";

$eventType = '_x_INFO'; $eventDesc = "procOrder R.$Grevision EXIT OK - USER: $Guser/$GcfPath CLIENT: $GclientName  opDSN ($totalRecords): $LAYOUT{'orderPr
ocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thisMailingTable'}  ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\*
FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

close(FH_out); close(FH_exc); close(FH_log); close(FH_report); close(FH_excReport);close(FH_excBODYReport);

exit;

################################################
################### SUBS #####################
################################################

#
# -------------------------------------------------------------------
#

sub alphaOrdered {

my(@singleElementArray) = @_;

my $alpha       = 0;
my $i           = 0;

my $base = $singleElementArray[0];

for($i = 1; $i <= $#singleElementArray; $i++) {

        $ordBase = ord($base); $ordArr = ord($singleElementArray[$i]);

##      print "==debug=$i= base = |$base| ordBase= $ordBase ___ singleElementArray = $singleElementArray[$i] ordArr = $ordArr \n";

        if($ordBase > $ordArr) { return $alpha; }
        $base = $singleElementArray[$i];

}

$alpha = 1;

return $alpha;

}

#
# -------------------------------------------------------------------
#
sub canonicalize {

my($inRecord) = @_;
my($elementsIndex, $fieldsIndex, $rec, $buf, $field) = undef;
my(@field) = undef;
my $space = ' ';
my $dashEscape = $space.'_d_esc_'.$space;

$inRecord = uc($inRecord);

my @allFields = undef;

@allFields = split(/$GfieldDelimiter/,$inRecord);
my $fieldsIndex = undef;

for($fieldsIndex=0; $fieldsIndex <= $#allFields; $fieldsIndex++) {

        if ($fieldsIndex != $emailFieldPos && $fieldsIndex != $fromAddressFieldPos) {

                if( $fieldsIndex ==  $companyFieldPos ||  $fieldsIndex == $lastNameFieldPos || $fieldsIndex == $firstNameFieldPos || $fieldsIndex == $titl
eFieldPos || $fieldsIndex == $countryFieldPos || $fieldsIndex == $departmentFieldPos  || $fieldsIndex == $cityFieldPos)  { $allFields[$fieldsIndex] =~ s/\
-/$dashEscape/g; }

                $allFields[$fieldsIndex] =~ s/\s+$//;                           # trailing space
                $allFields[$fieldsIndex] =~ s/^\s+//;                           # leading space
                $allFields[$fieldsIndex] =~ s/[\"\`\,\\\/\(\)]/$space/g;        # extraneous chars
                $allFields[$fieldsIndex] =~ s/\s{2,}/$space/g;                  # more than one separating space

        }
                elsif ($fieldsIndex == $emailFieldPos || $fieldsIndex != $fromAddressFieldPos) { $allFields[$fieldsIndex] =~ s/$space//g; }

        @field = split($space,$allFields[$fieldsIndex]);

        $field = undef;

        for($elementsIndex = 0; $elementsIndex <= $#field; $elementsIndex++) {

                CONVERT_TITLE_CASE: {

                $leaveUpper = 1;

                        if($fieldsIndex == $postalCodeFieldPos) { last CONVERT_TITLE_CASE; }
                        if($headerFields[$fieldsIndex] !~ /\w*IX\w*$/ && $field[$elementsIndex] =~ /^[^AEIOU\.][^AEIOU\.][^AEIOU\.]?$/) { last CONVERT_TIT
LE_CASE; }
                        if($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.[A-Z]?\.?$/)   { last CONVERT_TITLE_CASE; }

                        unless($field[$elementsIndex] =~ /(AND)|(OR)|(NEW)|(UND)|(AT)|(CO)|(LTD)|(INC)|(OF)|(THE)|(RD)/) {

                                if($fieldsIndex == $firstNameFieldPos && ($field[$elementsIndex] =~ /^[A-Z][A-Z]$/ || $field[$elementsIndex] =~ /^[A-Z]\.?
[A-Z]\.[A-Z]?$/)) { last CONVERT_TITLE_CASE; }
                                if(($fieldsIndex == $IXFieldPos ||  $fieldsIndex == $companyFieldPos)  && ($field[$elementsIndex] =~ /^[A-Z]\.?[A-Z]\.?[A-
Z]$/ || $field[$elementsIndex] =~ /^[I][IV][I]*$/ || $field[$elementsIndex] =~ /^[^JS]\.?[A-Z]\.?$/ )) { last CONVERT_TITLE_CASE; }
                                if($fieldsIndex == $titleFieldPos && $field[$elementsIndex] =~ /^[ARHMGIVUCS][EMSQ\&TRAFIOPV][PADO\.]*$/ && $field[$elemen
tsIndex] !~ /[MD][RS]/)     { last CONVERT_TITLE_CASE; }
                                if($fieldsIndex == $countryFieldPos && ($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.?[A-Z]?\.?$/ || $field[$elementsIndex] =
~ /^[A-Z][A-Z][A-Z]?$/) ) { last CONVERT_TITLE_CASE; }

                        }

                        unless($fieldsIndex == $stateFieldPos && length($field[$elementsIndex])>5) { if($fieldsIndex == $stateFieldPos || $fieldsIndex ==
$postalCodeFieldPos)   { last CONVERT_TITLE_CASE; } }

                $leaveUpper = 0;


                        $buf = lc($field[$elementsIndex]);

                        if($fieldsIndex != $emailFieldPos && $fieldsIndex != $fromAddressFieldPos)  { $buf = ucfirst($buf); last CONVERT_TITLE_CASE; }
                }

        if($leaveUpper) { $buf = $field[$elementsIndex]; }

        $field .= ($buf.$space);
        }

$rec .= ($field.$GfieldDelimiter);
}

chop($rec);

$rec =~ s/$dashEscape/\-/g;

return $rec;
}

```

```perl
#
# ------------------------------------------------------------------------------------------
#

sub checkDupsReturnMap {

my($ptrAOHDB,$GDEBUG)    = @_;
my(@DB)                  = @$ptrAOHDB;

## prepare map of duplicates on FromAddress

my %dupsMap              = undef;
my $j                    = undef;

for $j (0 .. $#DB) {

        $dupsMap{$j} = $DB[$j]{'FromAddress'}

}

my @mirrorDupsMap        = undef;
@mirrorDupsMap           = values(%dupsMap);
my $key                  = undef;
my $value                = undef;
my @arr                  = undef;
my $number               = undef;

foreach $key(keys %dupsMap) {

        $value           = $dupsMap{$key};
        next if($value !~ /\S+?/);

        @arr             = undef;
        @arr             = grep /$value/, @mirrorDupsMap;
        $number          = scalar(@arr);

        if($number > 1) {

                $eventType      = 'INFO'; $eventDesc = "DUPLICATE FromAddress ITEM DETECTED ON IMPORT & FLAGGED: ($number|$key|$value)"; &logEvent(\*FH_lo
g, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
                $dupsMap{$key}  = ('###'.$number.'### '.$dupsMap{$key});

        ## print $dupsMap{$key}."\n";

        }
        ## print "** key=".$key." *** value=".$dupsMap{$key}."___occurrences=".$number."\n";
}

return %dupsMap
}

#
# -----------------------------------------------------------------------
#

sub checkMultiBuy_Extract {

my($thisLine,$thisMailingID,$pos,$ptrLAYOUT,$GDEBUG) = @_;
my($exit) = 0;

my(%LAYOUT)                        = %$ptrLAYOUT;
my(@rangeStack)                    = undef;
my(@singleStack)                   = undef;
my($countRangeMatches)             = 0;
my($repeat)                        = 1;
my($match)                         = undef;
my($replacementEscChar)            = '%';
my($preProcRemoveChar)             = '^';
my($elementSeparator)              = ' ';
```

```

## PREPARE THE LINE FOR MATCHING

$thisLine                        = uc($thisLine);

## RUN RULESET FROM RULES CF FILE

foreach(@GPreProcRules) { eval; }

my $thisPreExtractionLine        = $thisLine;
my @testSingleStack              = undef;
my $maxASCII                     = 0;

## EXTRACTION LOOP

while($repeat) {

my $RULE = 0;
my $match = undef;
##print "\n\n--debug-- maxASCII = $maxASCII -- thisLine IS CURRENTLY |$thisLine| \n\n";

        MATCH_CASE: {

                SINGLE_CASE: {

##                      if($thisLine =~ /^([A-Z])$/)                                              { $match = $1; $RULE = -1;  last MATCH_CASE; }
                        if($thisLine =~ /^([A-Z])[^A-Z\^\%\@\-]{1}/)                              { $match = $1; $RULE = -2;  last MATCH_CASE; }
                        if($thisLine =~ /[^A-Z\^\%\@\-]([A-Z])[^A-Z\%\^\@\-]{1}[^A-Z\^\%\@\-]*/)   { $match = $1; $RULE = -3;  last MATCH_CASE; }

                }

                RANGE_CASE: {

                        if($thisLine =~ /^([A-Z]\-[A-Z])[^A-Z\^\%\@]{1}/)                          { $match = $1; $RULE = 1; last MATCH_CASE; }
                        if($thisLine =~ /[^A-Z\%\^\@]{1}([A-Z]\-[A-Z])[^A-Z\^\%\@]{1}/)            { $match = $1; $RULE = 2; last MATCH_CASE; }
                        ## if($thisLine =~ /([A-Z][A-Z][A-Z]*)[^A-Z\^\%]{1}/)                      { $match = $1; $RULE = 3; last MATCH_CASE; }
                        ## if($thisLine =~ /[^A-Z]{1}([A-Z][A-Z][A-Z]*)/)                          { $match = $1; $RULE = 4; last MATCH_CASE; }
                }



        $repeat = 0;
        }

        $thisLine =~ s/$match/$replacementEscChar/g;

        $thisCheckedMailingID = ($thisMailingID!=0) ? $thisMailingID : -1;

                ##print "\n- debug - ELEMENT CONSIDERED with break = $RULE |$match| \n";
                if($GDEBUG) { print "--debug-- BEGIN RULES -- RANGE = |$LAYOUT{$thisCheckedMailingID}| from $thisCheckedMailingID \n"; }

## BEGIN RULES TO SEE WHAT WE GOT ##

        $PUSH = 1;

        if($RULE > 0) {

                my(@arr) = undef;
                my $tempMatch = $match;
                $tempMatch =~ s/\-//g;
                @arr = split(//,$tempMatch);

                if(!&alphaOrdered(@arr)) {

                        ## print "-- debug -- REJECT - not alpha match |$match| in line |$thisPreExtractionLine| \n";
                        $PUSH = 0;

                }

        }

        #if($RULE > 0 && ord($arr[0]) <= $maxASCII) {

        #       print "-- debug -- REJECT - range case |$match| out of alpha order with current stack!\n";
        #       if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
        #       $PUSH = 0;

        #}

        if(defined($match) && $match !~ /[$LAYOUT{$thisCheckedMailingID}]/) {

                ## print "--debug-- REJECT - not in RANGE = |$LAYOUT{$thisCheckedMailingID}| match |$match| in line |$thisPreExtractionLine|  \n" ;
                if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
                $PUSH = 0;

        }

        if($PUSH && $RULE > 0 && $match =~ /\S+?/) {

##              push(@rangeStack,  ('<'.$RULE.'> |'.$match)); $exit = 1;
                $match =~ s/\-/ thru /g;
                push(@rangeStack, $match);
                $exit=1;
        }

        if($PUSH && $RULE < 0 && $match =~ /\S+?/) {

                push(@testSingleStack,$match);

                unless(ord($match) <= $maxASCII) {

##                      push(@singleStack, ('<'.$RULE.'> |'.$match));
                        push(@singleStack,$match);
                        $exit=1;

                }
                        if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
        }

                if($GDEBUG) { if(!defined($match)) {print "--debug-- match not defined! in line= |$thisPreExtractionLine|\n"; }}
}

## consolidate extractions, pre report

foreach(@rangeStack)    { unless($_ !~ /\S+?/) { push(@thisItemsSelected,$_); }}
foreach(@singleStack)   { unless($_ !~ /\S+?/) { push(@thisItemsSelected,$_); }}

## SPECIAL POST EXTRACTION CASE RULES

        my $FIND_BUY_ALL = 0;

        foreach(@GBuyAllIndicators) { s/[\n\r]//g; next if($_ !~ /\S+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_BUY_ALL = 1; } }

        if ($FIND_BUY_ALL) {

                if($exit) {

                        my $t = '.vfyTok';
                        $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS ALL-ITEM SELECTION DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

                push(@thisItemsSelected," ALL ($thisMailingID) "); $exit=1;
        }

        ## DETECT DIFFERENCE IN EXTRACTION STACK AND MAX EXTRACTION STACK

        if ($exit && (scalar(@singleStack) != scalar(@testSingleStack)) ) {

                my $t = '.vfyTok';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t:  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING / INSPECT IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType
 , $eventDesc, $Guser, $LGDEBUG_FTN);

        }

        my $FIND_ADD_CHANGE = 0;

        foreach(@GAddressChangeIndicators) { s/[\n\r]//g; next if($_ !~ /\S+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_ADD_CHANGE = 1; } }

        if ($FIND_ADD_CHANGE) {

                my $t = '.chkAC';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType,
 $eventDesc, $Guser, $LGDEBUG_FTN);

        }

        my $FIND_QUESTION = 0;

        if($thisPreExtractionLine =~ /\?\s/) { $FIND_QUESTION = 1; }

        if ($FIND_QUESTION) {

                my $t = '.chkQst';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE OR CARE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
entType, $eventDesc, $Guser, $LGDEBUG_FTN);

        }

        my $FIND_AI_ONLY = 0;

        if ($exit && scalar(@singleStack) == 2 && scalar(@testSingleStack) == 2) {

        ## $x = scalar(@singleStack);
        ## $y = scalar(@testSingleStack);

        ##print "-- debug -- AI TEST ___ singleStackLen = $x __ testSingleStackLen = $y \n";

                if(grep /(A)|(I)/, @singleStack) { $FIND_AI_ONLY = 1;}

        }

        if ($FIND_AI_ONLY) {

                my $t = '.vfyTok';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: \"A\" or \"I\" ONLY DETECTED"; &logEvent(\*FH_log, $eventType, $eventD
esc, $Guser, $LGDEBUG_FTN);

        }

        my $LIKELY_PARSE_ERRORS = 0;

        if ($exit && $thisPreExtractionLine =~ /[\[\]]+/) { $LIKELY_PARSE_ERRORS = 1; }

        if ($LIKELY_PARSE_ERRORS) {

                my $t = '.vfyTok';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: BRACKETS FOUND IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
entType, $eventDesc, $Guser, $LGDEBUG_FTN);
```

```

        }


        ##print "\n-$thisRecordID- debug singleStack -- \n";
        ##print @singleStack;
        ##print "\n-- debug testSingleStack -- \n";
        ##print @testSingleStack;
        ##print "\n-$thisRecordID---------------------------\n";


return $exit;
}

#
# ---------------------------------------------------------------------------
#

sub detectBuyTokens {

my($MULTI_BUY_ON,$Gsubj_buyToken,$thisSubject,$thisMailingID,$thisBodyTop,$thisBodyBottom,$ptrHASHLAYOUT,$ptrSCALARtopLines,$ptrSCALARbottomLines,$GDEBUG)
 = @_;

my %LAYOUT               = %$ptrHASHLAYOUT;
my @topLines             = @$ptrSCALARtopLines;
my @bottomLines          = @$ptrSCALARbottomLines;
my $buyTokenPosition     = 'none';
$thisBuyStatus           = undef;
@thisItemsSelected       = undef;
$thisItemsSelectedStack = undef;

my $exit = 0;

## print "--debug-- buyToken= |$Gsubj_buyToken|\n";

FIND_BUY_TOKENS: {

##
## &checkMultiBuy_Extract returns success on a good match/extraction;
## @thisItemsSelected is created/populated at this time as well
##

## check $thisSubject

## print "\n\n==========debug-- SUBJECT = |$thisSubject| \n";

        if(!$MULTI_BUY_ON && $thisSubject =~ /$Gsubj_buyToken/) {

                $buyTokenPosition = ' S '; $thisBuyStatus = ('<'.$buyTokenPosition.'> |'.$thisSubject.'|');

## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";

        last FIND_BUY_TOKENS;
        }

        if($MULTI_BUY_ON) {

                if(&checkMultiBuy_Extract($thisSubject,$thisMailingID,' S ',\%LAYOUT,$GDEBUG)) {

                        $buyTokenPosition = ' S '; $thisBuyStatus = '<'.$buyTokenPosition.'>';

                last FIND_BUY_TOKENS;
                }

        }

## check @topLines
## print "\n\n==========debug-- thisBodyTop = |$thisBodyTop| \n";
```

```
        if(!$MULTI_BUY_ON) {

                foreach(@topLines) {

                        $thisTopLine = $_;

                        if($thisTopLine =~ /$Gsubj_buyToken/) {

                                $buyTokenPosition = ' T ';
                                $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisTopLine;

## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";

                        }

                }

        last FIND_BUY_TOKENS;
        }


        if($MULTI_BUY_ON) {

                if(&checkMultiBuy_Extract($thisBodyTop,$thisMailingID,' T ',\%LAYOUT,$GDEBUG)) {

                        $exit =1;

                        $buyTokenPosition = ' T ';
                        $thisBuyStatus = '<'.$buyTokenPosition.'> ';

                last FIND_BUY_TOKENS;
                }

        }


## check @bottomLines

## print "\n\n==========debug-- thisBodyBottom = |$thisBodyBottom| \n";

        if(!$MULTI_BUY_ON) {

                foreach(@bottomLines) {

                        $thisBottomLine = $_;

                        if($thisBottomLine =~ /$Gsubj_buyToken/) {

                                $buyTokenPosition = ' B ';
                                $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisBottomLine;

## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";

                        }

                }

        last FIND_BUY_TOKENS;
        }

        if($MULTI_BUY_ON) {

                if(&checkMultiBuy_Extract($thisBodyBottom,$thisMailingID,' B ',\%LAYOUT,$GDEBUG)) {

                        $exit =1;

                        $buyTokenPosition = ' B ';
                        $thisBuyStatus = '<'.$buyTokenPosition.'> ';
                last FIND_BUY_TOKENS;
                }

        }

}

## print "--debug-- ==== $i ====== buyTokenPosition= |$buyTokenPosition|\n";

return $buyTokenPosition;
}

#
# ----------------------------------------------------------------------
#

sub extractDedupItemsSelectedStack {

my(@thisItemsSelected) = @_;

my $thisItemsSelectedStack = undef;
my $thisItemsSelectedStackSeparator = ',';
my $i = undef;

for($i = 0; $i <= $#thisItemsSelected; $i++) {

        my $item = $thisItemsSelected[$i]; # all items must be normalized!

        next if($item !~ /\S+?/);

        $item=&normalizeSpacing($item);

        $thisItemsSelectedStack .= ($item.$thisItemsSelectedStackSeparator);

}

$thisItemsSelectedStack =~ s/\,\s*$//;

return  $thisItemsSelectedStack;
}

#
# ----------------------------------------------------------------------
#

sub getDateStamp {

# get standardized date stamp (UNIX, WINNT)
# returns a date stamp

my($Gplatform,$LGDEBUG_FTN) = @_;

if(!defined($Gplatform)) { $Gplatform = 'WIN'; }
my $GTIME       = '';
my $day         =  undef;
my $val         =  undef;

if ($Gplatform ne 'WIN') { $val = `date '+%y%m%d%H%M%S-%a'`; }
        else {
                $val = `date /t`;
                chop($val);
                $val =~ /(\w+)\s+(\d+)\/(\d+)\/(\d{2})(\d{2})/;
                $val = $5.$2.$3; $day = $1;
        }

if ($Gplatform eq 'WIN') {

                $GTIME = `echo '' | time`;
                $GTIME =~ /\:\s*(\d{1,2})\:(\d{2})\:(\d{2})\.(\d{2})/;
                my $leadingDigit = $1;
                if (length($leadingDigit) == 1) { $leadingDigit = ('0'.$leadingDigit); }
                $val .= $leadingDigit.$2.$3.$4.'-'.$day;
        }

if($LGDEBUG_FTN) { print "DEBUG__gdate=".$val."\n"; }

return $val;

}

#
# ----------------------------------------------------------------------
#

sub getFieldKeys {

my($ptrLayout) = @_;
my(%LAYOUT) = %$ptrLayout;

## prepare and count required fields from body data layout for check against body

my(@fieldKeys)  = undef;
my($k)          = undef;

##$f = scalar(@fieldKeys); print "--debug-- scalarStartFtn=$f \n";

foreach $k(sort keys %LAYOUT) {

        next if($k !~ /^[a-z]{1,2}_/);  ## these elements are not fieldnames but are other CF parameters (name \t value)

        ## print "--debug-- k= |$k|\n";

        if($k =~ /\S+?/) { push(@fieldKeys,$k); }
}

shift @fieldKeys;
## $f = scalar(@fieldKeys); print "--debug-- scalarEndFtn=$f \n";

return @fieldKeys;
}

# ------------------------------------------------------------------------------------------

sub getKeyPos_Outfile {

my($ptrHASHrecordTemplate) = @_;

my(%HASHrecordTemplate) = %$ptrHASHrecordTemplate;

my @record                      = undef;
$salutationFieldPos     = -1;
$IDFieldPos             = -1;
$LIDFieldPos            = -1;
$MailingIDFieldPos      = -1;
$firstNameFieldPos      = -1;
$lastNameFieldPos       = -1;
$IXFieldPos             = -1;
$titleFieldPos          = -1;
$stateFieldPos          = -1;
$countryFieldPos        = -1;
$postalCodeFieldPos     = -1;
$emailFieldPos          = -1;
$companyFieldPos        = -1;
$addressFieldPos        = -1;
$departmentFieldPos     = -1;
$cityFieldPos           = -1;
$fromAddressFieldPos    = -1;

```

```perl
my $i                   = 0;

@record = sort(keys(%HASHrecordTemplate));

##print "---DEBUG---"; print @record; sleep 4;

for($i=0; $i <= $#record; $i++) {

        $record[$i] = uc($record[$i]);

        CASE: {

                if ($record[$i] =~ /FROM\_ADDRESS/)                                                     { $fromAddressFieldPos  = $i; last CASE; }
                if ($record[$i] =~ /(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNET_AD)|(INTERNETAD)/)       { $emailFieldPos        = $i; last CASE; }
                if ($record[$i] =~ /(COUNTRY)|(CTRY)/)                                                  { $countryFieldPos      = $i; last CASE; }
                if ($record[$i] =~ /(ZIP)|(POSTAL)/)                                                    { $postalCodeFieldPos   = $i; last CASE; }
                if ($record[$i] =~ /(INST)|(COMPANY)|(ORGANIZATION)/)                                   { $companyFieldPos      = $i; last CASE; }
                if ($record[$i] =~ /DEPARTMENT/)                                                        { $departmentFieldPos   = $i; last CASE; }
                if ($record[$i] =~ /(ADDR)|(ADDRESS)/)                                                  { $addressFieldPos      = $i; last CASE; }
                if ($record[$i] =~ /CITY/)                                                              { $cityFieldPos         = $i; last CASE; }
                if ($record[$i] =~ /(CUSTNO)/)                                                          { $IDFieldPos           = $i; last CASE; }
                if ($record[$i] =~ /(LISTID)/)                                                          { $LIDFieldPos          = $i; last CASE; }
                if ($record[$i] =~ /(PRIORITY)|(MAILING)/)                                              { $MailingIDFieldPos    = $i; last CASE; }
                if ($record[$i] =~ /(.*F\w+NAME.*)|(FIRST)/)                                            { $firstNameFieldPos    = $i; last CASE; }
                if ($record[$i] =~ /(.*L\w+NAME.*)|(LAST)/)                                             { $lastNameFieldPos     = $i; last CASE; }
                if ($record[$i] =~ /^.*IX.*$/)                                                          { $IXFieldPos           = $i; last CASE; }
                if ($record[$i] =~ /TITLE/)                                                             { $titleFieldPos        = $i; last CASE; }
                if ($record[$i] =~ /STATE/ || $record[$i] =~ /^\_C$/)                                   { $stateFieldPos        = $i; last CASE; }
        }
}

if ($firstNameFieldPos  == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\+F\\w\+NAME\/ was not found!\n"; }
if ($lastNameFieldPos   == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\+L\\w\+NAME\/ was not found!\n"; }
if ($firstNameFieldPos  == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\*IX\\w\*NAME\/ was not found!\n"; }
if ($titleFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/TITLE\/ was not found!\n"; }
if ($stateFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/STATE\/ was not found!\n"; }
if ($postalCodeFieldPos == -1)          { print STDOUT "[HEADER] WARNING: \/ZIP|POSTAL\/ was not found!\n"; }
if ($countryFieldPos    == -1)          { print STDOUT "[HEADER] WARNING: \/(COUNTRY)|(CTRY)\/ was not found!\n"; }
if ($emailFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNET_AD)|(INTERNETAD)\/ was not found!\n"; }

}


#
# ---------------------------------------------------------------------
#

sub getNextDBkey_return {

my($keyFileName,$LGDEBUG_FTN) = @_;

$LOCK_SH = 1;
$LOCK_EX = 2;
$LOCK_NB = 4;
$LOCK_UN = 8;

$POS_BOF = 0;
$POS_CUR = 1;
$POS_EOF = 2;

$/ = undef;
$startKeys = 0;

if(! -s $keyFileName) { `echo $startKeys >> $keyFileName`; }

OPEN_DB: if (!open(KEYS, "+< $keyFileName")) {$exit = 0; $eventType = 'FAIL'; $eventDesc = "getNextDBkey_return::Cannot open: \[$keyFileName \]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}

```

```perl
flock(KEYS, $LOCK_EX);

        my($count)      = <KEYS>;
        $nextKey        = $count+1;
        seek(KEYS,0,$POS_BOF);
        print KEYS $nextKey;

flock(KEYS, $LOCK_UN);
close(KEYS);

return $nextKey;
}

#
# ---------------------------------------------------------------------
#

sub HASHloadFieldMap {

my($fileName,$LGDEBUG_FTN) = @_;

if (!open(FH, "$fileName")) { $eventType = 'FAIL'; $eventDesc = "HASHloadFieldMap::Cannot open: \[$keyFileName \]"; &logEvent(\*FH_log, $eventType, $event
Desc, $Guser, $LGDEBUG_FTN);}

$/ = "\n";

my(@fieldMapTemp) = <FH>;
my($key) = undef;
my($value) = undef;
my(%fieldMap) = undef;

close(FH);

        foreach (@fieldMapTemp) {

                my $item = $_;

                next if(($item =~ /\/\//) || ($item !~ /\S+?/)); # comments

                ($key, $value) = split(/[\t\,]/,$item);

                $value =~ s/[\n\r]//g;

                $fieldMap{$key} = $value;

        }

return %fieldMap;
}

#
# ---------------------------------------------------------------------
#

sub loadFile_SCALAR {

my($fileName) = @_;

if (!open(FILE, "$fileName")) {$eventType = 'FAIL'; $eventDesc = "loadArray::Cannot open fileName: \[$fileName\]"; &logEvent(\*FH_log, $eventType, $eventD
esc, $Guser, $LGDEBUG_FTN);}

$/ = "\n";

my(@all) = undef;

while(<FILE>) {

        my $buf = $_;
        next if($buf !~ /\S+?/ || $buf =~ /^\/\//);
        push(@all, $buf);

}

return @all;
}

#
# ----------------------------------------------------------------------
#

sub logEvent {

my($FH,$eventType,$eventDesc,$Guser,$LGDEBUG_FTN) = @_;

my $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
# print event to STDOUT

if($eventType eq 'ABORT') {

        $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$Guser $GclientName $eventType $eventDesc"`;
        die "Error! Cannot log! ".$eventType."\t".$eventDesc."\n" ;
}

if($eventType !~ /_x_/) { print $eventType."\t".$eventDesc."\n"; }

        else { $eventType =~ s/_x_//; }

# write event to activity log

        &writeRecord(\*FH_log,$GfieldDelimiter,$thisLogKey=&getNextDBkey_return($GactivityLogKeysFileName,$LGDEBUG_FTN),\%GactivityLogRecord,'',$LGDEBUG_FTN);
        if($eventType eq 'FAIL') {

                $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$Guser $GclientName $eventType $eventDesc"`;
                exit;

        }
}

#
# ----------------------------------------------------------------------
#

sub normalizeSpacing {

my($dirty) = @_;

$dirty =~ s/\s{2,}/$space/g; $dirty =~ s/\s+$//; $dirty =~ s/^\s+//;
$clean = $dirty;

return $clean
}

#
# ----------------------------------------------------------------------
#

sub queryUPDATE_DB {

my($ptrLAYOUT,$thisQueryKey, $thisQueryFieldName, $fieldName, $fieldValue, $fieldName1, $fieldValue1, $RECORD_RECOVERED, $LGDEBUG_FTN) = @_;

##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";

my(%LAYOUT) = %$ptrLAYOUT;
my $RS = undef;
my $Conn = undef;
my $Errors = undef;
my @fieldNames = undef;
my $PID = -1;
my $recCount = 0;
my $EXIT = 1;

# construct/open DSN

my $Conn = undef;

$Conn = new Win32::OLE("ADODB.Connection");
$Conn->Open($LAYOUT{'thisMailingDSN'},"","");

my(%DB) = undef;

# Load data into recordset

        my $mailingTable = $LAYOUT{'thisMailingTable'};
        my %foo = undef;

# look to see if update record exists, $thisQueryKey

        unless($RECORD_RECOVERED) {

                my $RECORD_EXISTS = 1;
                %foo = &querySELECT_DB(\%LAYOUT,$thisQueryFieldName,$thisQueryKey,$LGDEBUG_FTN);
                if($foo{'_exit_'} < 1) { $RECORD_EXISTS = 0; $EXIT = 0; return $EXIT; }
        }

        $thisQuery = "UPDATE $mailingTable SET $fieldName = \'$fieldValue\', $fieldName1 = \'$fieldValue1\'  WHERE $thisQueryFieldName = \'$thisQueryKey\'
";

        $RS = $Conn->Execute($thisQuery);

##%foo = %$RS;
##foreach(keys(%foo)) { print "---key = $_ __ value = $foo{$_} \n"; }

        if(!$RS) {

                $Errors = $Conn->Errors();
                print "Errors:\n";
                foreach $error (keys %$Errors) {
                        print $error->{Description}, "\n";
                }
                $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB'); $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) queryUPDATE_DB :: Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

        $EXIT = 0;
        return $EXIT;
        }

        $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) queryUPDATE_DB :: UPDATE SUCCESSFUL: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

return $EXIT;
}

#
# ----------------------------------------------------------------------
#

sub querySELECT_DB {

my($ptrLAYOUT,$thisQueryFieldName,$thisQueryKey,$LGDEBUG_FTN) = @_;

##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";

my(%LAYOUT) = %$ptrLAYOUT;
my $RS = undef;
my $Conn = undef;
my $Errors = undef;
my @fieldNames = undef;
my $PID = -1;
my $recCount = 0;

# construct/open DSN

my $Conn = undef;

$Conn = new Win32::OLE("ADODB.Connection");
$Conn->Open($LAYOUT{'thisMailingDSN'},"","");

my(%DB) = undef;
$DB{'_exit_'} = 1;

# Load data into recordset

my $mailingTable = $LAYOUT{'thisMailingTable'};

$thisQuery = "SELECT * FROM $mailingTable WHERE $thisQueryFieldName = \'$thisQueryKey\'";

$RS = $Conn->Execute($thisQuery);

if(!$RS) {

        $Errors = $Conn->Errors();
        print "Errors:\n";
        foreach $error (keys %$Errors) {
                print $error->{Description}, "\n";
        }
        $eventType = 'FAIL'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) querySELECT_DB :: Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
}

my $i = undef;

for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }

my $RESULT_FLAG = 1;

if($RS->EOF) { $RESULT_FLAG = 0; }
$RS -> MoveNext;

if(!$RS->EOF) { $RESULT_FLAG = -1; }
$RS -> MovePrevious;

if($RESULT_FLAG < 1) {

        (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB'); $FUPcount++; $eventType = 'ERR_FUP'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) querySELECT_DB :: Bad Lookup ($RESULT_FLAG): \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

        $DB{'_exit_'} = $RESULT_FLAG;

return %DB;
}

while ( !$RS->EOF ) {

        for($i=0; $i < $RS ->Fields->Count; $i++) {

                $DB{$fieldNames[$i]}= uc($RS->Fields->Item($i)->value);
        }

        $RS->MoveNext; $recCount++;
}

$Conn->Close;

```

```
return %DB;
}

#
# ------------------------------------------------------------------------
#

sub read_DSNtoAOH {

my($DSN,$LGDEBUG_FTN) = @_;

##print "** DSN = $DSN \n";

$RS = undef;
$Conn = undef;
$Errors = undef;
@fieldNames = undef;
$PID = -1;
$recCount = 0;

# construct/open DSN

$Conn = new Win32::OLE("ADODB.Connection");
$Conn->Open($DSN,"","");
my(@FILE) = undef;

# Load data into recordset

$thisQuery = "SELECT * FROM Email";

$RS = $Conn->Execute($thisQuery);

if(!$RS) {
        $Errors = $Conn->Errors();
        print "Errors:\n";
        foreach $error (keys %$Errors) {
                print $error->{Description}, "\n";
        }
        $eventType = 'FAIL'; $eventDesc = "read_DSNToAOH::Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_
FTN);
}

my $i = undef;

for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }

while ( !$RS->EOF ) {

        for($i=0; $i < $RS ->Fields->Count; $i++) {

         $FILE[$recCount]{$fieldNames[$i]}= $RS->Fields->Item($i)->value;

        }

        $RS->MoveNext; $recCount++; if ($recCount % 20 == 0) { print STDOUT "+"; }
}

$Conn->Close;

return @FILE;
}

#
# ------------------------------------------------------------------------
#

sub removeTabs {

my($tabs) = @_;
```

```

$tabs = s/\t//g;

return $tabless

}

#
# ----------------------------------------------------------------------
#

sub subjectSynonymnLookup {

my($subject,$ptrHASHLAYOUT) = @_;
my(%LAYOUT) = %$ptrHASHLAYOUT;
my($thisMailingID) = -1;

        $subject =~ /\s*([\=\*\+\-]{1,3})\s*/;
        $thisSubjectMailingIDToken = $1;

        if(defined($LAYOUT{$thisSubjectMailingIDToken})) { $thisMailingID = $LAYOUT{$thisSubjectMailingIDToken}; }

        ## print "-- debug -- ATTEMPT MAILING ID RECOVERY... TOKEN= |$thisSubjectMailingIDToken|... RESULT MAILINGID= |$thisMailingID|\n";

return $thisMailingID
}

#
# ----------------------------------------------------------------------
#

sub writeRecord {

# takes FH by ref; has _e_ feature for values

my($FH,$GfieldDelimiter,$logKey,$ptrHASHrecord,$RECORD_TYPE, $LGDEBUG_FTN) = @_;

flock($FH, $LOCK_EX);

my $fileEmpty = 0;
if(-z $FH) {$fileEmpty = 1;}

%HASHrecord = %$ptrHASHrecord;

$space = " ";
$exit = 1;

$LOCK_SH = 1;
$LOCK_EX = 2;
$LOCK_NB = 4;
$LOCK_UN = 8;

my $record = undef;
my $header = undef;
my $key = undef;

foreach $key(sort(keys(%HASHrecord))) {

        $value  = $HASHrecord{$key};

        if($fileEmpty) { $key =~ s/[a-z]{1,2}_//; $header .= (uc($key).$GfieldDelimiter); }

        if($value =~ /^_e_(.{2,})$/) { $value = eval($1); }


        $record .= ($value.$GfieldDelimiter);

}
chop($record);                                          # remove last field delimiter
```

```

if($fileEmpty) { chop($header); print $FH $header."\n"; }
if($ADD_CHANGE_ON && $RECORD_TYPE ne 'BODY') { $record = &canonicalize($record); }
print $FH $record."\n";

flock($FH, $LOCK_UN);
}

__END__

```